CAM
Computational & Applied Mathematics

Hierarchical multilevel methods for exascale UQ and optimization

Challenges and Possible paths forward

Clayton Webster[⋆] & **Stefan Wild**[†]

[⋆]Department of Computational & Applied Mathematics (CAM)
Oak Ridge National Laboratory, and

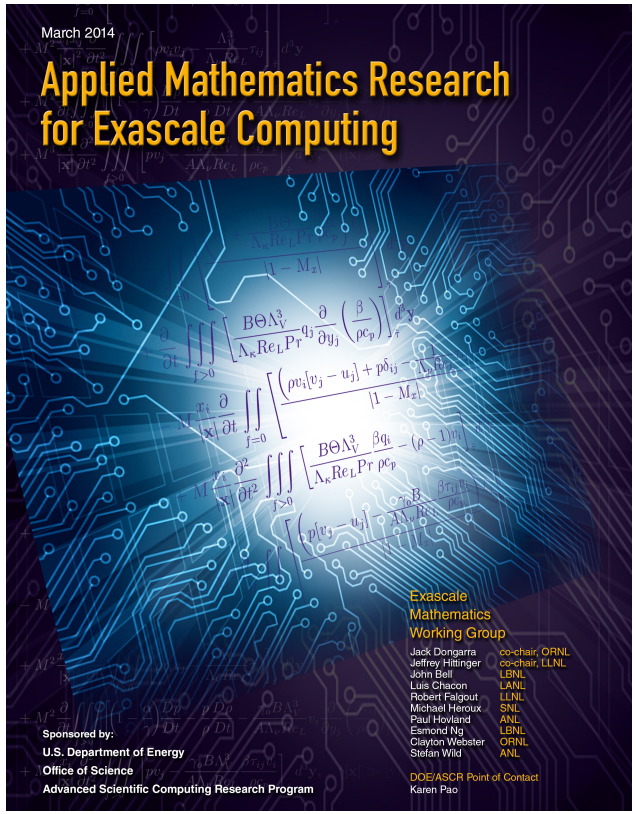
Department of Mathematics, University of Tennessee

[†]Computer Science and Mathematics Division
Argonne National Laboratory

July 11, 2014

Outline

- See Section 4.2.2 and 4.2.3 in the *Applied Mathematics Research for Exascale* →



From petascale to exascale

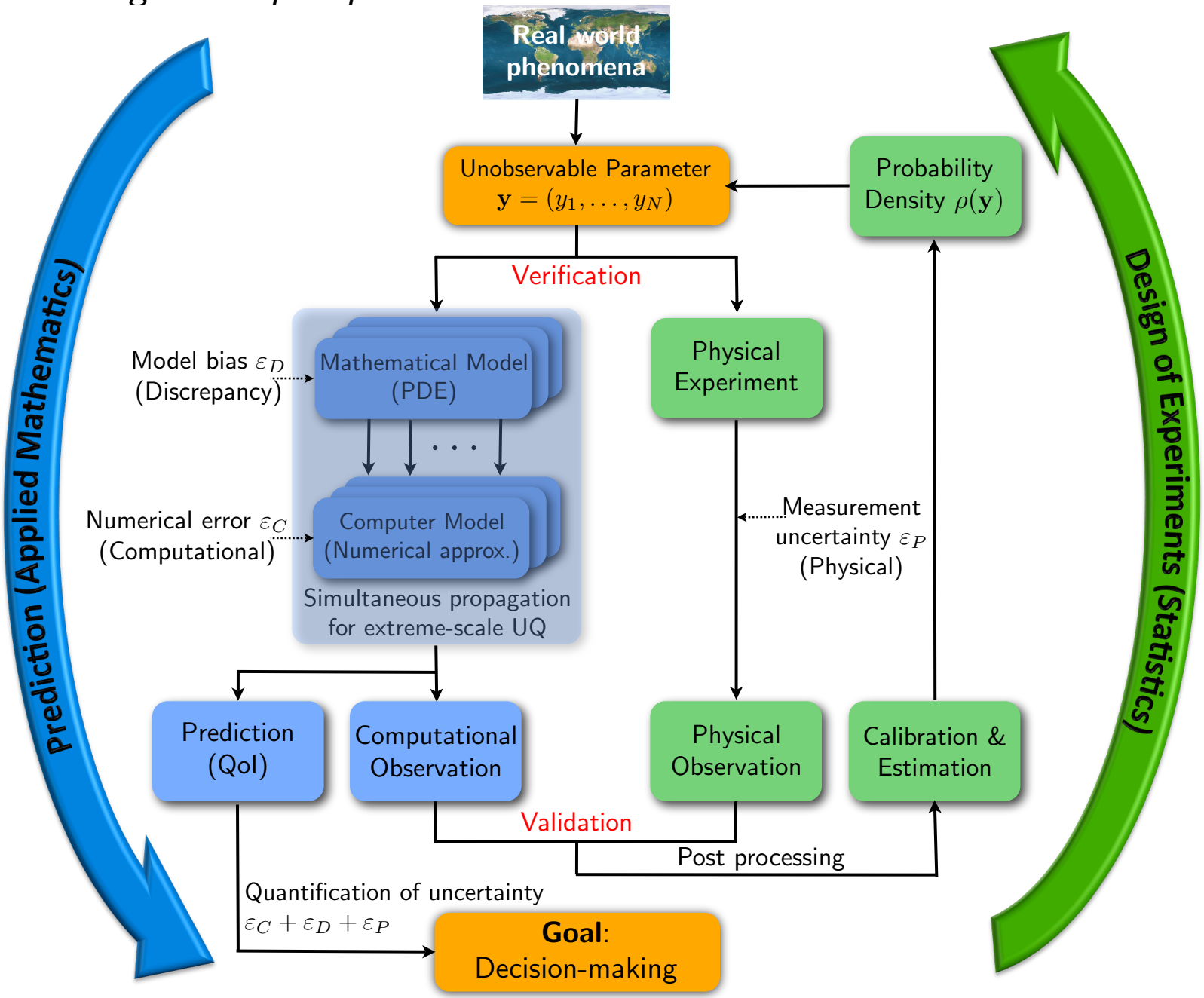
Some challenges remain but new challenges (and opportunities) emerge

CAM Computational & Applied Mathematics

In moving towards exascale, several challenges arise when applying UQ methodologies to the DOE mission science areas.

1. Detection and quantification of high-dimensional stochastic QoIs with a specified certainty
2. Reducing the computational burden required to perform rigorous UQ
3. Efficient strategies for UQ that exploit greater levels of parallelism provided by emerging many-core architectures
4. Systematic assimilation of the uncertainty in measured data for validating and correcting model bias, calibrating parameter interrelations, and improving confidence in predicted responses

EQUINOX: *An architecture-aware, predictive capability for explaining how the uncertainties, ubiquitous in all modeling efforts affect our predictions and understanding of complex phenomena*



Forward UQ: PDEs with random input data

Complexity reduction for uncertainty quantification

CAM Computational & Applied Mathematics

parameters $\boldsymbol{y}(\omega)$, $\omega \in \Omega_{\mathbb{P}}$ $\longrightarrow$ **SPDE model:** $\mathcal{L}(u, \boldsymbol{y}) = f$ for a.e. $x \in D \subset \mathbb{R}^d$ $\longrightarrow$ quantity of interest $Q[u(\cdot, \boldsymbol{y})]$

- The parameters $\boldsymbol{y}(\omega)$ may be affected by uncertainty (experimental data, incomplete description of parameters, unresolved scales, etc.)
- $\boldsymbol{y} : \Omega \to \Gamma \subset \mathbb{R}^N$ can be assumed to be a random vector with N components, i.e., $\boldsymbol{y} = (y_1, \ldots, y_N)$, with joint probability density function $\rho(\boldsymbol{y})$

The solution u is a stochastic function, $u(\cdot, \boldsymbol{y})$

Goals of forward UQ: Approximate u or some statistical QoI depending on u, i.e.

$$\mathbb{E}[u],\ \mathbb{V}\mathrm{ar}[u],\ \mathbb{P}[u > u_0] = \mathbb{E}[\mathbb{1}_{\{u>u_0\}}]$$

with as minimal computational cost as possible

Brief taxonomy of (current) numerical strategies

Stochastic FEMs [Gunzburger-W-Zhang, Acta Numerica 2014]

- Let $\mathcal{H}_M(\Gamma) = \{\boldsymbol{y}_m \in \Gamma\}_{m=1}^M$ denote a set of (possible random) sample points
- Let $u_h(x, \boldsymbol{y}_m) := u_h(\boldsymbol{y}_m)$ for $m = 1, \ldots, M$ denote the finite element approximation to the parametric PDE on a fixed mesh h

Monte Carlo methods

$$\mathbb{E}[u] \approx E_M[u_h] = \frac{1}{M} \sum_{m=1}^{M} u_h(x, \boldsymbol{y}_m)$$

pro: convergence rate is independent of N

con: asymptotic rate is $\mathcal{O}(1/\sqrt{M})$

Stochastic polynomial methods

- Stochastic Galerkin: projection technique, intrusive approach
- Stochastic collocation: interpolation technique, non-intrusive approach

$$u \approx u_{M,h}^{(SL)} = \sum_{m=1}^{M} c_m(x)\psi_m(\boldsymbol{y})$$

pro: convergence can be faster than MC

con: curse of dimensionality

- $\{\psi_m\} \in \mathcal{P}(\Gamma)$ polynomial basis and c_m determined through, e.g., $u_h(\boldsymbol{y}_m)$

Scalability constraints of current UQ

Näive implementations yield suboptimal efficiency

Exascale will require
... billion-way parallelism
... with less memory per core
... at potentially lower clock speeds

Current UQ approaches (both non-intrusive and intrusive) provide massive parallelism, but they don't scale (well) now and **won't** scale in the future:

- Independent simulation $\rightarrow$ easy parallelism...
  ... but independent instantiations $\rightarrow$ poor memory resource usage.
  This approach eats memory as fast as it fills compute cores

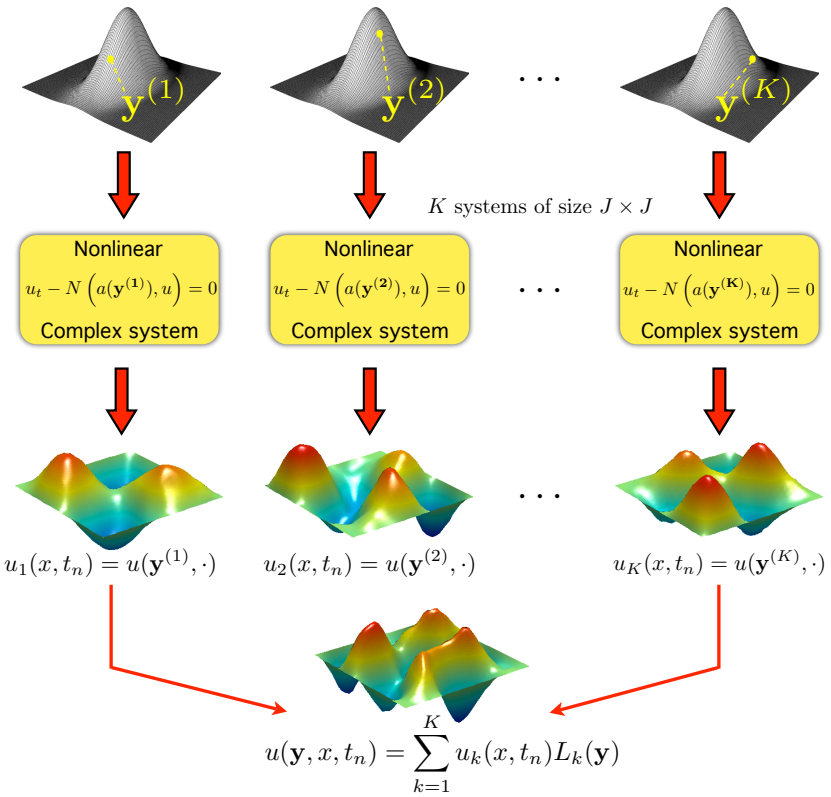
- Independent solves means that similarity between systems cannot be exploited for improved efficiency

- Process-level parallelism is simple, but finer-grained parallelism (thread and vector) is superior

This näive approach is ultimately too restrictive

An embedded UQ paradigm

Simultaneous propagation on ensembles



Key Idea: Simultaneous propagation of samples through a single program instantiation, allowing:

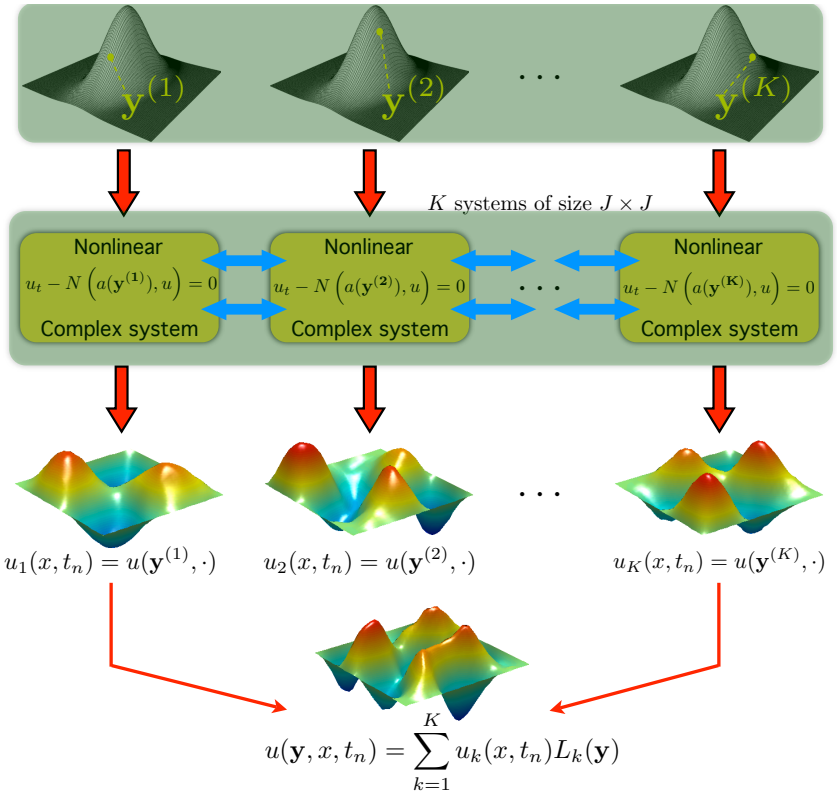
- block solvers to exploit related systems to accelerate solver convergence
- finer-grained parallelism to yield speedup even for a single core
- shared-memory parallelism to permit sharing common data and reducing contention for memory resources

Result: faster time-to-solution, independent of parallel speedup (esp. important if power/reliability require slower clock speeds) and reduced memory resources

These benefits are significant on current architectures, and will be crucial for future extreme-scale architectures

An embedded UQ paradigm

Simultaneous propagation on ensembles



Key Idea: Simultaneous propagation of samples through a single program instantiation, allowing:

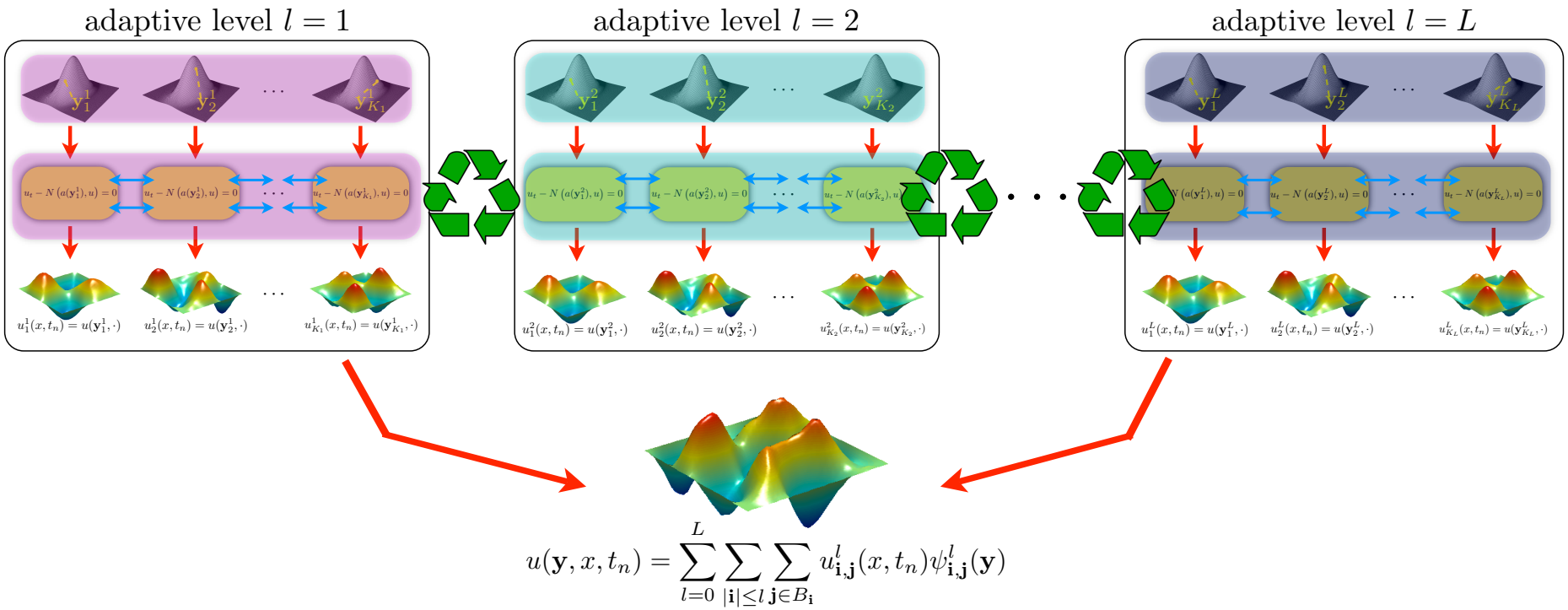
- block solvers to exploit related systems to accelerate solver convergence
- finer-grained parallelism to yield speedup even for a single core
- shared-memory parallelism to permit sharing common data and reducing contention for memory resources

Result: faster time-to-solution, independent of parallel speedup (esp. important if power/reliability require slower clock speeds) and reduced memory resources

These benefits are significant on current architectures, and will be crucial for future extreme-scale architectures

Multilevel hierarchical UQ methods

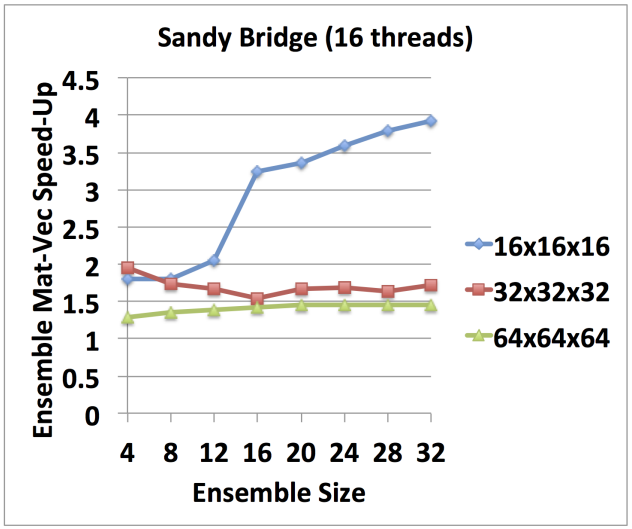
Reduce, reuse and recycle - exploit the hierarchies!

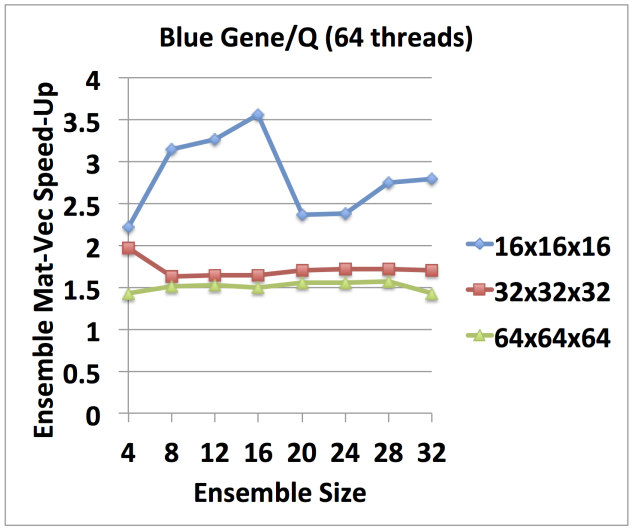


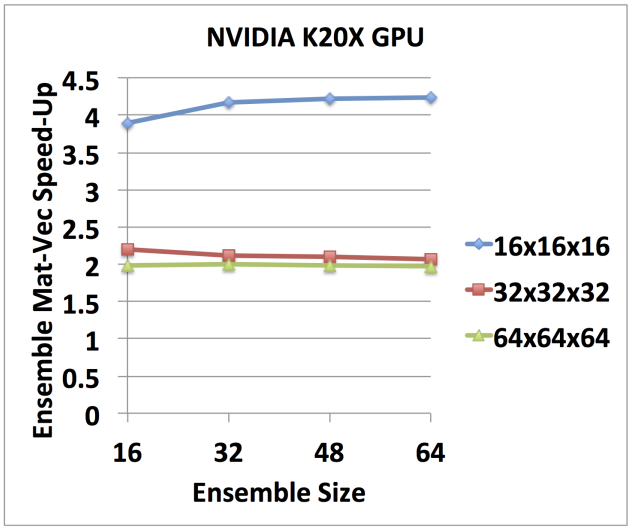
- Exploit similarities between block system across the hierarchical adaptive levels of both the deterministic and stochastic approximations
- *Embedded UQ* approach permits recycling Krylov subspace solvers, providing another opportunity to reduce time-to-solution and memory consumption through reduced solver iterations

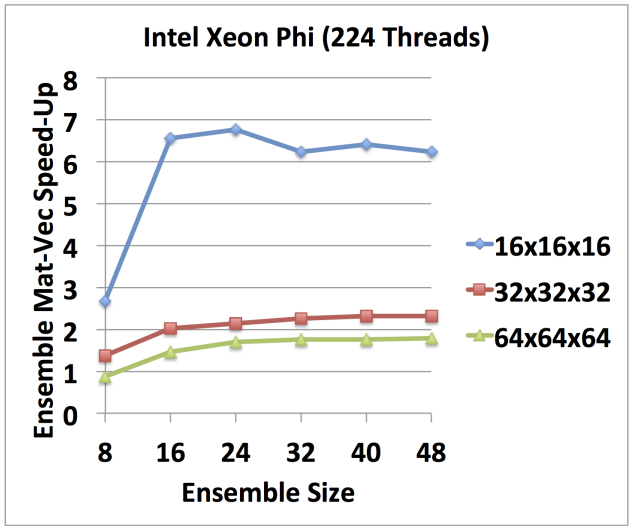
Embedded UQ methods

Ensemble AMG-preconditioned CG speed-up



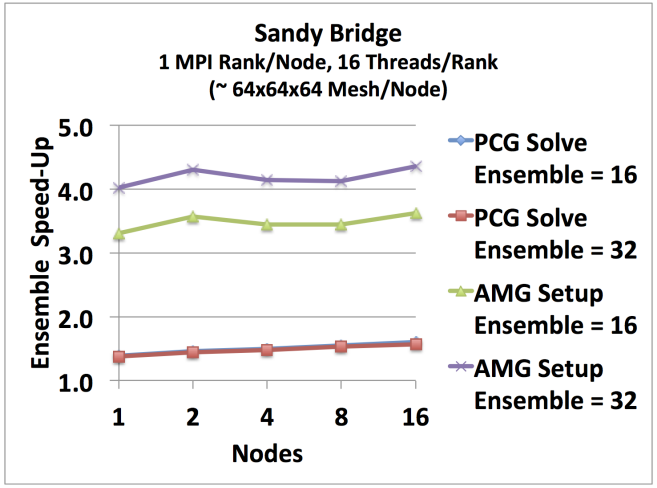


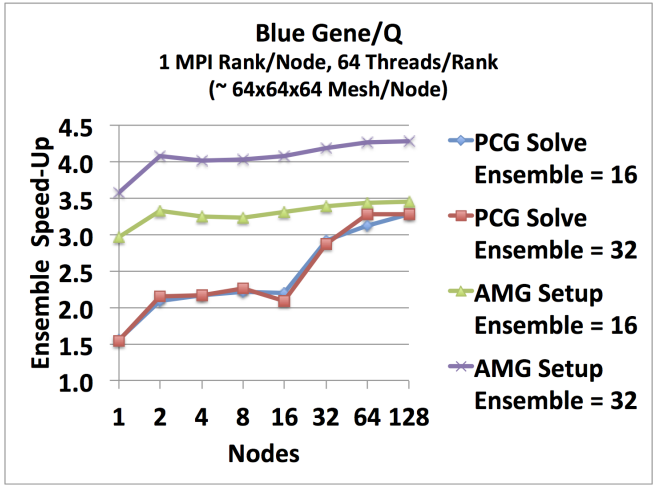


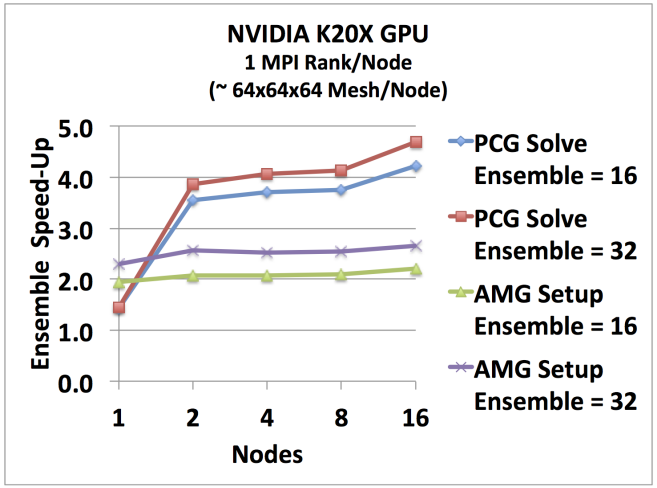


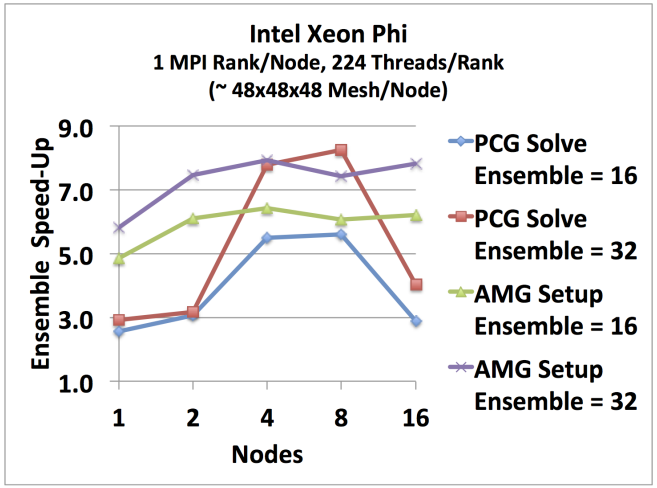
Embedded UQ methods

Ensemble matrix-vector product speed-up









Hierarchical stochastic collocation (HSC) methods

Complexity reduction through solution acceleration

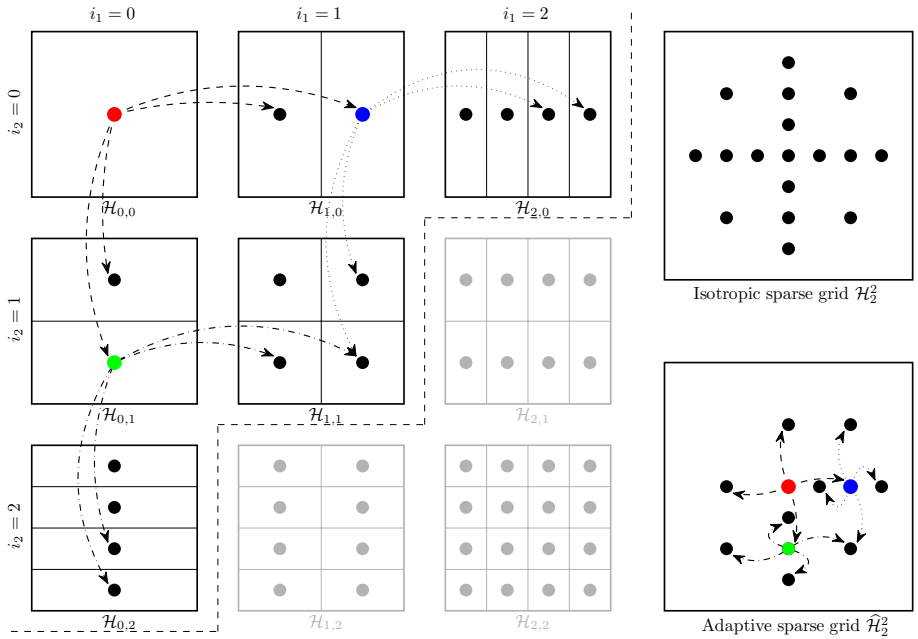
At level $L \in \mathbb{N}_+$, a hierarchical SC approximation is (informally) defined by:

$$u_{M_L,h} = \mathcal{I}_{M_L}[u_h] \equiv \mathcal{I}_{M_{L-1}}[u_h] + \Delta_L[u_h]$$

- $\mathcal{I}_{M_{L-1}}[u_h]$ is the hierarchical interpolant at level $L-1$, and $\Delta_L[u_h]$ is the corresponding hierarchical surplus interpolant at level L

An approach for reducing complexity - exploit the stochastic hierarchy

Construct lower level interpolants $u_{M_{L-1},h}(x, \widetilde{\boldsymbol{y}}_k)$ for $\{\widetilde{\boldsymbol{y}}_k\} \in \Delta\mathcal{H}_{M_L}$ as an initial guess for $u_{M_L,h}(x, \boldsymbol{y})$, to accelerate the underlying (iterative) deterministic solvers

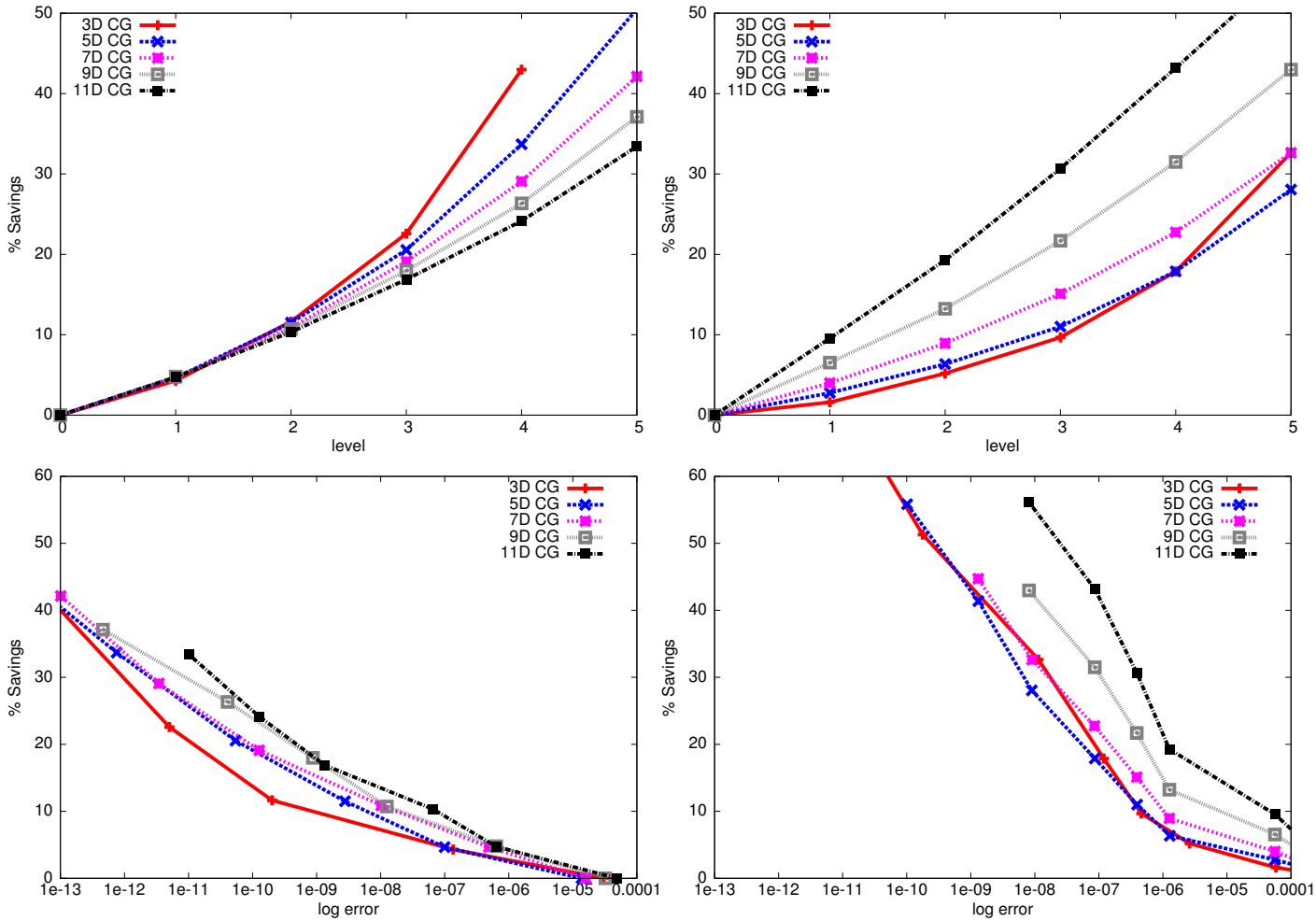


[Gunzburger-W-Zhang, 2013, Jantsch-Galindo-W-Zhang, 2014]

Computational savings of HSC methods

QoI $= \mathbb{E}[u]$ for nonlinear elliptic SPDEs (CG iterative solver)

Figure Savings versus level and savings versus error for $L = 1/64$ (left) and $L = 1/2$ (right)

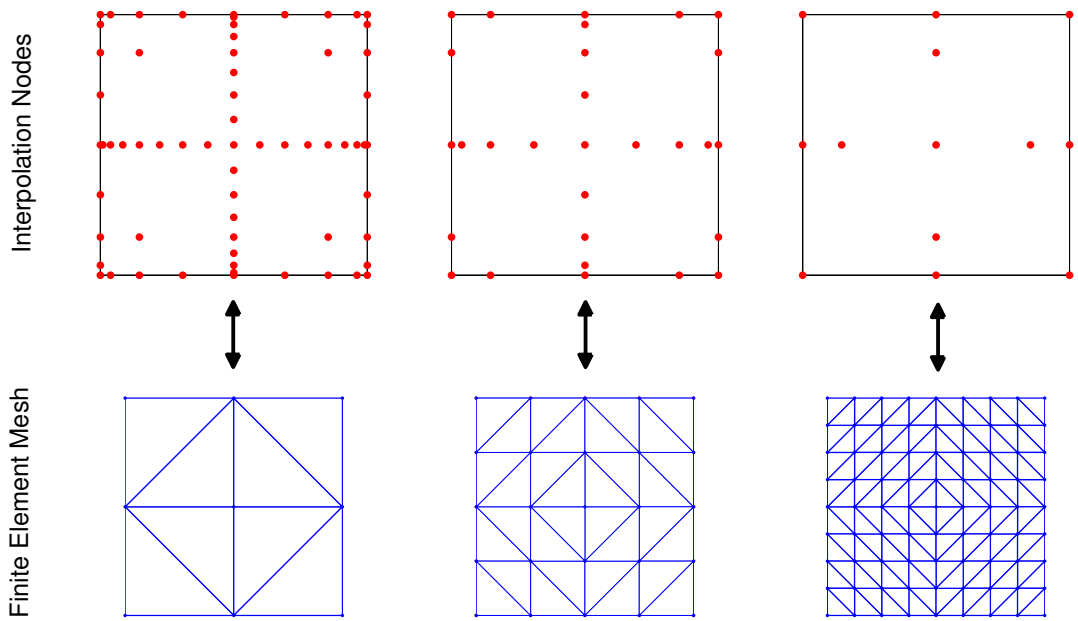


Multilevel stochastic collocation (MLSC) methods

Exploit the deterministic hierarchy [Gunzburger-Jantsch-Teckentrup-W, 2014]

Basic idea: As in the single level case, to increase convergence (compared to MLMC), we simply interpolate the differences $u_{h_k} - u_{h_{k-1}}$ at different resolutions

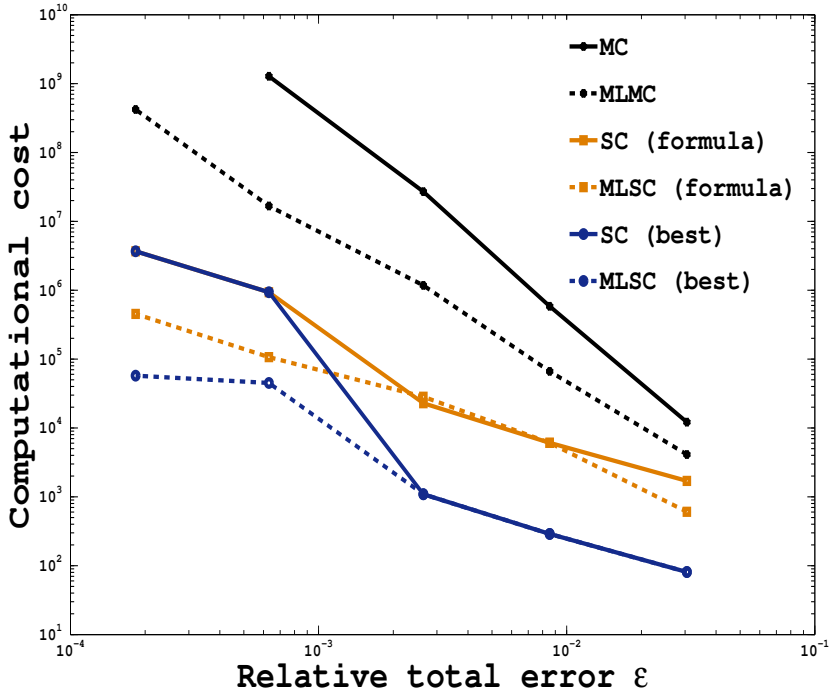
$$u_K^{(MLSC)} = \sum_{k=0}^{K} \mathcal{I}_{M_{K-k}} \left[u_{h_k} - u_{h_{k-1}} \right] = \sum_{k=0}^{K} \left(u^{(\mathrm{SL})}_{M_{K-k},h_k} - u^{(\mathrm{SL})}_{M_{K-k},h_{k-1}} \right)$$



- For a given accuracy, multilevel methods seek to reduce complexity by spreading computational cost evenly across several resolutions of the spatial discretization

Computational savings of MLSC methods

Results in 10D



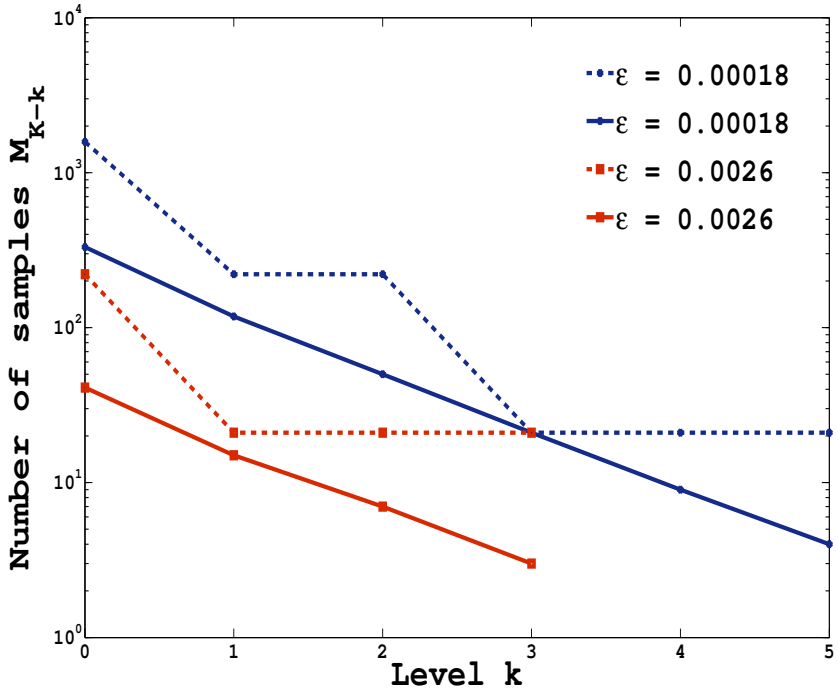
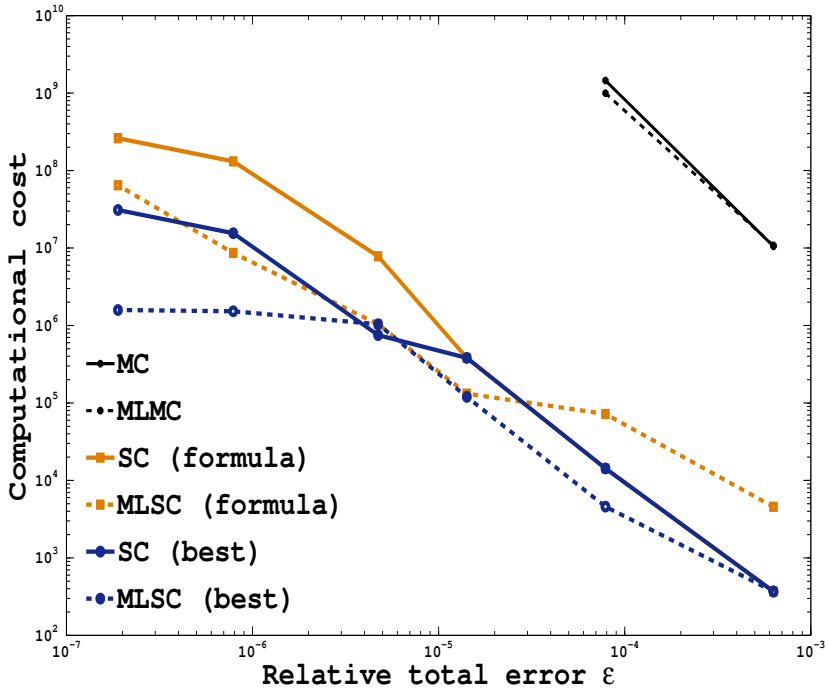


Figure Left: Cost versus Error for $D = (0, 1)^2$, $N = 10$. Right: Number of samples per level (predicted vs actual).

Computational savings of MLSC methods

Results in 20D



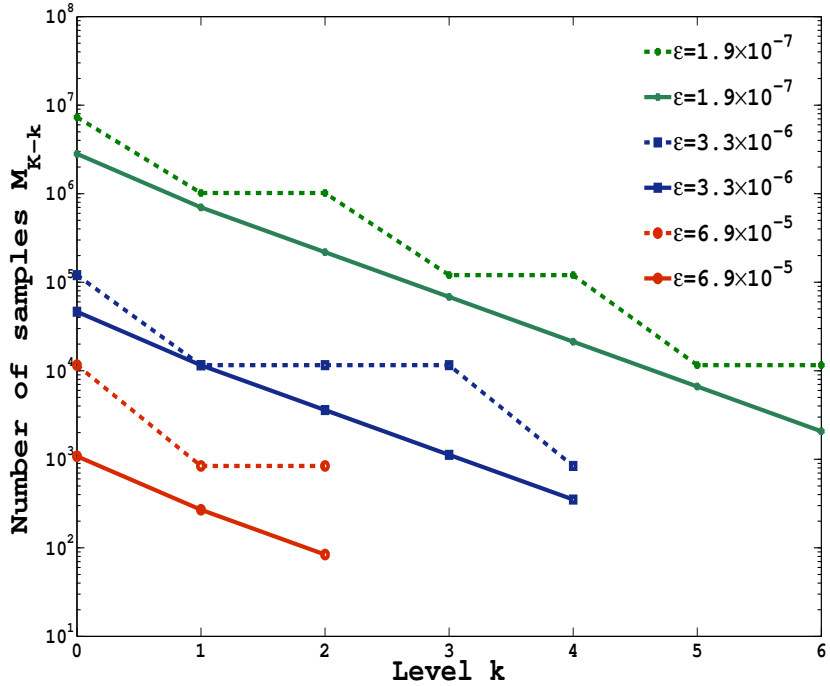


Figure Left: Cost versus Error for $D = (0, 1)$, $N = 20$. Right: Number of samples per level (predicted vs actual).

Optimization and Exascale Computing

CAM Computational & Applied Mathematics

Expect exascale computing to:

- Enable consideration of optimal design for new application areas
- Drive new (at this scale/difficulty level) optimization problems
  - Mixed-integer PDE-constrained optimization
  - Global optimization
  - Robust optimization, Optimization under uncertainty
- Require fundamentally different algorithmic approaches
  - Breaking outer optimization loop-inner simulation separation
  - Concurrent/multi-point function-derivative-subproblem evaluations
  - Multifidelity/hierarchical methods
  - Algorithm-based fault tolerance

New Paradigms

Mixed-Integer PDE-Constrained Optimization Under Uncertainty

CAM Computational & Applied Mathematics

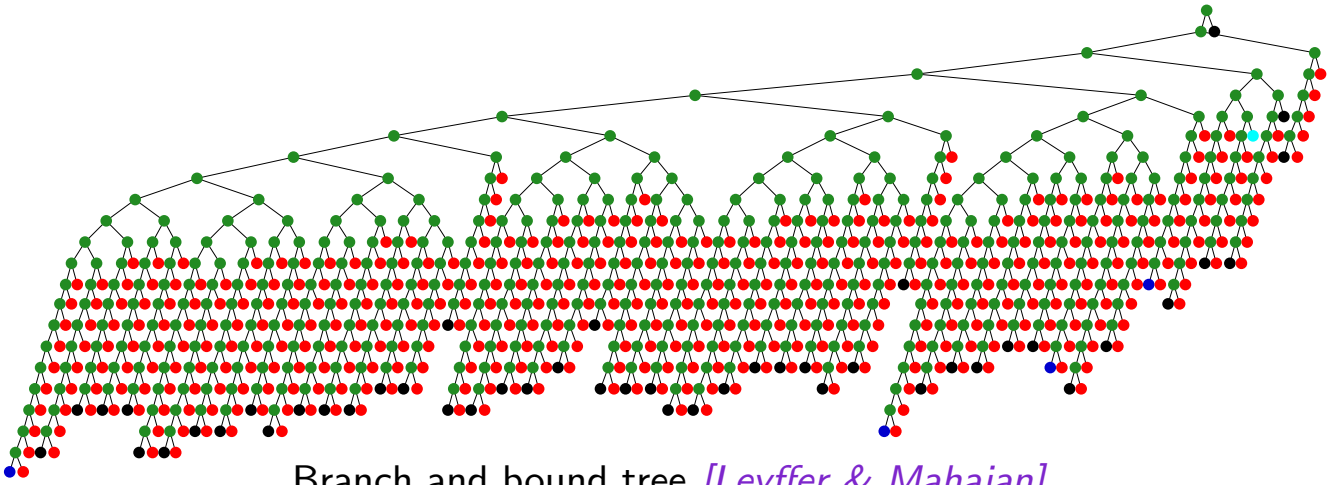
$$\min_{\mathbf{x},\boldsymbol{y}(\omega),\mathbf{z}} \{\mathbb{E}_\omega \left[f(\mathbf{x};\boldsymbol{y}(\omega);\mathbf{z})\right] : \mathbf{c}(\mathbf{x};\boldsymbol{y}(\omega);\mathbf{z};\omega) = 0 \quad \text{a.s. in } \Gamma,\ \forall \mathbf{x} \in \mathcal{X},\ \mathbf{z} \in \mathbb{Z}_z^n \cap \mathcal{Z}\}$$

- $\mathbb{E}f$ Objective
- $\mathbf{x}$ Continuous design variables
- $\boldsymbol{y}(\omega)$ State variables depending on random variables $\omega \in \Omega$
- $\mathbf{z}$ Integer design variables
- $\mathbf{c}$ Linking constraints (PDE and boundary conditions)
- $\mathcal{X}, \mathcal{Z}$ Design constraints

New Paradigms

Sample Challenges/Opportunities

MIPDECO adds a combinatorial explosion to PDECO



Branch and bound tree *[Leyffer & Mahajan]*

- Each node is a PDECO solve
- Rethink today's PDECO (e.g., *[Biros & Ghattas: LNKS]*)
  - Integrate multilevel combinatorial with multilevel PDE
  - Map related PDE/PDECO solves to machine to exploit reuse
  - Allow for approximate/multifidelity PDE/PDECO solves

Mathematical/Numerical Optimization

Today's Generic Methods

Current iterate $\mathbf{x}^k \in \mathbb{R}^n$

1. Generate direction(s) $\mathbf{d}^k \in \mathbb{R}^n$
   - Ex- Newton(-Krylov) direction: $\mathbf{H}^k\mathbf{d}^k = (\approx) -\mathbf{g}^k$
     $\mathbf{H}^k \approx \nabla^2 f(\mathbf{x}^k)$ (or $\nabla^2 L(\mathbf{x}^k)$),
     $\mathbf{g}^k \approx \nabla f(\mathbf{x}^k)$ (or $\nabla L(\mathbf{x}^k)$)
   - Ex- Sequential quadratic programming:
     $\min_{\mathbf{d}\in\mathbb{R}^n} \left\{\mathbf{d}^T\mathbf{H}^k\mathbf{d} + \mathbf{d}^T\mathbf{g}^k : \nabla\mathbf{c}(\mathbf{x}^k)\mathbf{d} + \mathbf{c}(\mathbf{x}^k) = 0\right\}$
   - Ex- Stochastic: $-\left(\nabla f(\mathbf{x}^k)^T\mathbf{d}^k\right)\mathbf{d}^k$, random $\mathbf{d}^k$
2. Determine step length $\alpha^k > 0$
   - Ex- Line search: approximately solve
     $\min_{\alpha\geq 0} \phi(\mathbf{x}^k + \alpha\mathbf{d}^k)$ using merit function ϕ
   - Ex- Trust region: constrain $\|\alpha^k\mathbf{d}^k\| \leq \Delta^k$
   - Ex- Fixed step size: $\alpha_k = \kappa_k$
3. Update $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{d}^k$,
   evaluate $f(\mathbf{x}^{k+1})$, $\nabla f(\mathbf{x}^{k+1})$, ...

Inherently sequential iterations

- Typical focus: reduce work/time per iteration
- Appeal to Newton/Nesterov for fast convergence
- Evaluations of f, ∇f, $\mathbf{Hv}$, ... occur sequentially
- Requires global synchronization at each k (except for very special cases)

Mathematical/Numerical Optimization

Today's Generic Methods

CAM Computational & Applied Mathematics

Current iterate $\mathbf{x}^k \in \mathbb{R}^n$

1. Generate direction(s) $\mathbf{d}^k \in \mathbb{R}^n$
   - Ex- Newton(-Krylov) direction: $\mathbf{H}^k\mathbf{d}^k = (\approx) -\mathbf{g}^k$
     $\mathbf{H}^k \approx \nabla^2 f(\mathbf{x}^k)$ (or $\nabla^2 L(\mathbf{x}^k)$),
     $\mathbf{g}^k \approx \nabla f(\mathbf{x}^k)$ (or $\nabla L(\mathbf{x}^k)$)
   - Ex- Sequential quadratic programming:
     $\min_{\mathbf{d}\in\mathbb{R}^n} \left\{\mathbf{d}^T\mathbf{H}^k\mathbf{d} + \mathbf{d}^T\mathbf{g}^k : \nabla\mathbf{c}(\mathbf{x}^k)\mathbf{d} + \mathbf{c}(\mathbf{x}^k) = 0\right\}$
   - Ex- Stochastic: $-\left(\nabla f(\mathbf{x}^k)^T\mathbf{d}^k\right)\mathbf{d}^k$, random $\mathbf{d}^k$
2. Determine step length $\alpha^k > 0$
   - Ex- Line search: approximately solve
     $\min_{\alpha\geq 0} \phi(\mathbf{x}^k + \alpha\mathbf{d}^k)$ using merit function ϕ
   - Ex- Trust region: constrain $\|\alpha^k\mathbf{d}^k\| \leq \Delta^k$
   - Ex- Fixed step size: $\alpha_k = \kappa_k$
3. Update $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\mathbf{d}^k$,
   evaluate $f(\mathbf{x}^{k+1})$, $\nabla f(\mathbf{x}^{k+1})$, ...

Inherently sequential iterations

- Typical focus: reduce work/time per iteration
- Appeal to Newton/Nesterov for fast convergence
- Evaluations of f, ∇f, $\mathbf{Hv}$, ... occur sequentially
- Requires global synchronization at each k (except for very special cases)

Today's Parallel Methods

Single Point per Iteration

CAM Computational & Applied Mathematics

1. Parallelize linear algebra within iteration
   - Used to evaluate f, ∇f, ...
   - Matrix-free Jacobian-/Hessian-vector products

     Ex.- Toolkit for Advanced Optimization based on PETSc *[Munson et al]*

   Exa! Explosion of concurrency: optimization must take part

2. Multilevel optimization methods
   - Rely primarily on grid structure for variables

     Ex- *[Toint et al]*, *[MG/OPT: Lewis & Nash]*

   Exa! Does not fully account for architectural hierarchies, adaptivity, multiphysics, ...

3. Assume function (+derivative) evals succeed at demanded tolerance

   Exa! Pay price for demanding resilient computation

   Exa! Analysis for ABFT to enlarge classes of failures under which some form of convergence still ensured

Today's Parallel Methods

Single Point per Iteration

CAM Computational & Applied Mathematics

1. Parallelize linear algebra within iteration
   - Used to evaluate f, ∇f, ...
   - Matrix-free Jacobian-/Hessian-vector products

     Ex.- Toolkit for Advanced Optimization based on PETSc *[Munson et al]*

   Exa! Explosion of concurrency: optimization must take part

2. Multilevel optimization methods
   - Rely primarily on grid structure for variables

     Ex- *[Toint et al]*, *[MG/OPT: Lewis & Nash]*

   Exa! Does not fully account for architectural hierarchies, adaptivity, multiphysics, ...

3. Assume function (+derivative) evals succeed at demanded tolerance

   Exa! Pay price for demanding resilient computation

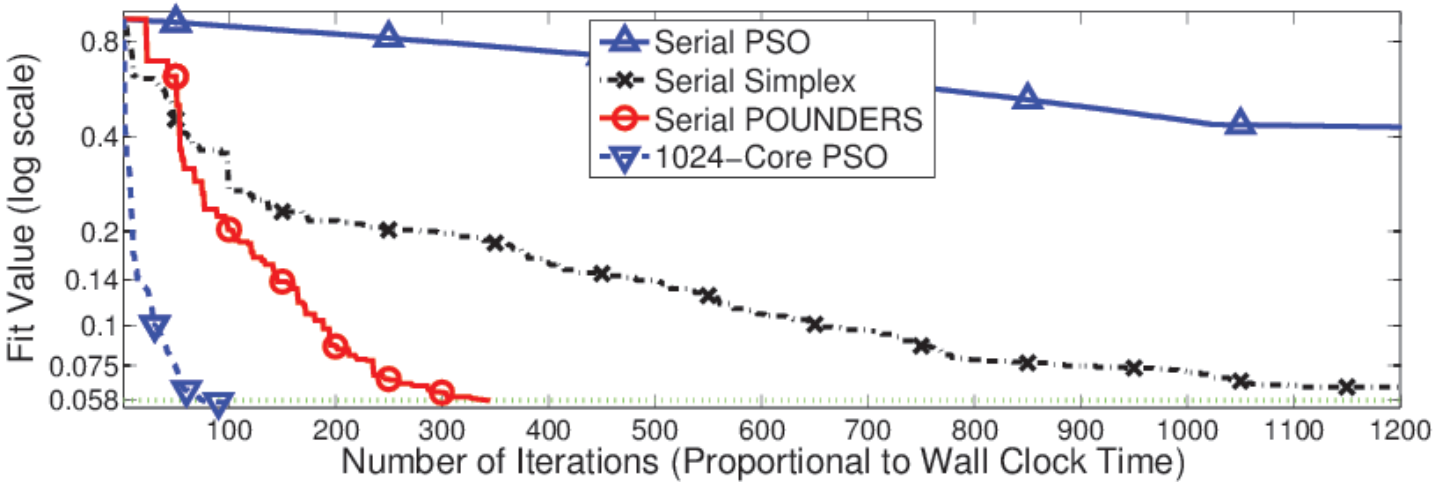
   Exa! Analysis for ABFT to enlarge classes of failures under which some form of convergence still ensured

Today's Parallel Methods

Concurrent Evaluations

- Primarily for derivative-free methods (e.g., $\nabla f(\mathbf{x})$ unavailable)
  - Ex- `HOPSPACK` *[Kolda et al]*: Evaluate $\{f(\mathbf{x}^k + \mathbf{d}_i) : i = 1, \ldots, p\}$ concurrently
  - Ex- `VTDIRECT` *[Watson et al]*: Concurrent evals for (approx) global optimization
  - Ex- `POUNDERS` *[W.]*: Evaluate residuals concurrently
  - Ex- Heuristics (GAs, particle swarm, ...): Concurrent generation evaluation

- Poor scaling of time to solution with respect to # of concurrent evals



  - Ex- `POUNDERS` (single evaluation) only 3 times slower than PSO (1024 concurrent evals) on accelerator design problem

Require New Methods and Analysis

Increased Concurrency for Derivative-Based Solvers?

Analysis to determine classes of problems where this works

- Krylov-based solutions to $\min_{\mathbf{x}\in\mathbb{R}^n}\{\|\mathbf{Ax}-\mathbf{b}\|\}$
  - s-step methods *[Chronopoulos 1991]*
  - CA/CH methods using matrix power kernel $\mathbf{Au}, \mathbf{A}^2\mathbf{u}, \ldots$
- Increase arithmetic intensity by evaluating local ensembles of related points
- Subspace/decomposition techniques
  - Trivial for separable problems ($\mathbf{x}_i \cap \mathbf{x}_j = \emptyset, i \neq j$):

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left\{\sum_{i=1}^{p} f_i(\mathbf{x}_i)\right\} \quad \longrightarrow \quad \min_{\mathbf{x}_i\in\mathbb{R}^{n_i}}\left\{f_i(\mathbf{x}_i)\right\},\ i=1,\ldots,p$$

  - More general: orthogonal/iterated subspaces + a synchronization step *[Gould et al, 1994]*, *[Yuan, 2007]*, *[Gratton et al, 2014]*

Decomposition to Reduce Outer Iterations

Back to the Future

Parallel variable distribution *[Ferris & Mangasarian, 1994]*

1. Generate partitioned direction $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_p) \in \mathbb{R}^n$, $\mathbf{d}_i \in \mathbb{R}^{n_i}$
   - Not block Jacobi/coordinate search because of $\mathbf{d}$
   - $\mathbf{d}$: Newton direction, steepest descent, ...
2. Concurrently solve p $(n_i + p - 1)$-dimensional subproblems

$$\min_{\mathbf{x}_i \in \mathbb{R}^{n_i}, \mathbf{u}_i \in \mathbb{R}^{p-1}} \{f(\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) : (\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) \in \mathcal{X}\}$$

   to obtain $\boldsymbol{y}^i = (\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) \in \mathbb{R}^n$

3. Obtain $\mathbf{x}^{k+1} = v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i$ from $(p+1)$-dimensional subproblem

$$\min_{\mathbf{v} \in \mathbb{R}^{p+1}} \left\{ f\left(v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i \right) : v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i \in \mathcal{X}, \quad \mathbf{v}^T\mathbf{e} = 1 \right\}$$

Decomposition to Reduce Outer Iterations

Back to the Future

Parallel variable distribution *[Ferris & Mangasarian, 1994]*

1. Generate partitioned direction $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_p) \in \mathbb{R}^n$, $\mathbf{d}_i \in \mathbb{R}^{n_i}$
   - Not block Jacobi/coordinate search because of $\mathbf{d}$
   - $\mathbf{d}$: Newton direction, steepest descent, ...
2. Concurrently solve p $(n_i + p - 1)$-dimensional subproblems

$$\min_{\mathbf{x}_i \in \mathbb{R}^{n_i}, \mathbf{u}_i \in \mathbb{R}^{p-1}} \{f(\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) : (\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) \in \mathcal{X}\}$$

   to obtain $\boldsymbol{y}^i = (\mathbf{x}_i, \mathbf{D}_{-i}\mathbf{u}_i) \in \mathbb{R}^n$

3. Obtain $\mathbf{x}^{k+1} = v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i$ from $(p+1)$-dimensional subproblem

$$\min_{\mathbf{v} \in \mathbb{R}^{p+1}} \left\{ f\left(v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i \right) : v_{p+1}\mathbf{x}^k + \sum_i v_i \boldsymbol{y}^i \in \mathcal{X}, \quad \mathbf{v}^T\mathbf{e} = 1 \right\}$$

Optimization Under Uncertainty

Example: Sample Average Approximation

Approximately solve stochastic program

$$\min_{\mathbf{x}\in\mathcal{X}\subset\mathbb{R}^n} \{\mathbb{E}_\omega [f(\mathbf{x};\omega)] : \mathbf{c}(\mathbf{x};\omega) = 0 \quad \forall\omega \in \Omega\}$$

by solving

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ \sum_{i=1}^{N} f(\mathbf{x};\omega_i) : \mathbf{c}(\mathbf{x};\omega_i) = 0,\ i = 1,\ldots,N \right\}$$

using the N scenarios $\omega_1, \ldots, \omega_N$

- Constraint/objective evaluation naturally parallelizable in scenarios
- Specific forms of f, $\mathbf{c}$, $\mathcal{X}$ enable scalable linear algebra
- ! Increased scenarios for exascale concurrency rarely (never?) useful
- ! Number of outer iterations (= global reductions) does not decrease as N grows

Optimization Under Uncertainty

Example: Sample Average Approximation

CAM Computational & Applied Mathematics

Approximately solve stochastic program

$$\min_{\mathbf{x}\in\mathcal{X}\subset\mathbb{R}^n} \{\mathbb{E}_\omega [f(\mathbf{x};\omega)] : \mathbf{c}(\mathbf{x};\omega) = 0 \quad \forall\omega \in \Omega\}$$

by solving

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ \sum_{i=1}^{N} f(\mathbf{x};\omega_i) : \mathbf{c}(\mathbf{x};\omega_i) = 0,\ i = 1,\ldots,N \right\}$$

using the N scenarios $\omega_1,\ldots,\omega_N$

- Constraint/objective evaluation naturally parallelizable in scenarios
- Specific forms of f, $\mathbf{c}$, $\mathcal{X}$ enable scalable linear algebra
- ! Increased scenarios for exascale concurrency rarely (never?) useful
- ! Number of outer iterations (= global reductions) does not decrease as N grows

Today's Parallel Methods

SAA at the Petascale

A special case: 2-stage programs

- f convex, quadratic
- C linear (equalities) with a recourse term for each scenario

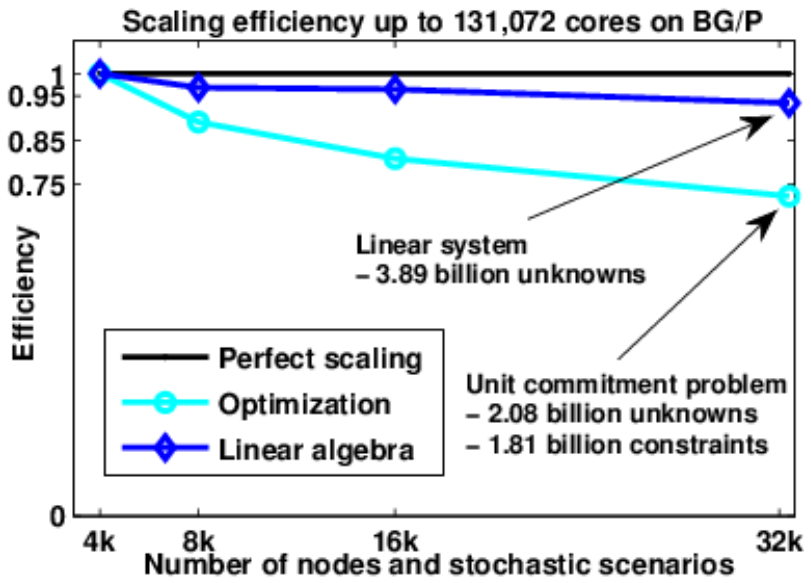
Interior point methods solve arrow systems

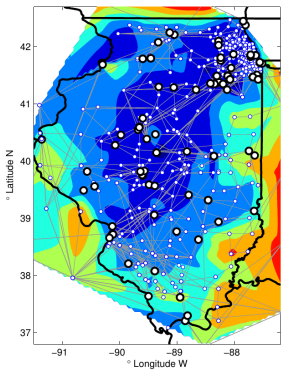
$$\begin{bmatrix} \mathbf{A}_1^k & & & \mathbf{B}_1^k \\ & \ddots & & \vdots \\ & & \mathbf{A}_N^k & \mathbf{B}_N^k \\ \mathbf{B}_1^{k\,T} & \cdots & \mathbf{B}_N^{k\,T} & \mathbf{A}_0^k \end{bmatrix} \delta^{k+1} = \mathbf{r}^k$$

in each iteration.

Dominant expense: Schur complements

$$\mathbf{B}_i^{k\,T} \left(\mathbf{A}_i^k\right)^{-1} \mathbf{B}_i^k$$





Solving energy unit commitment problems using PIPS

[Lubin, Petra, Schenck, Anitescu et al 2011–]

Concluding remarks

CAM Computational & Applied Mathematics

- Exascale optimization and uncertainty quantification methods must break the outer loop mentality:
  - exploitation of shared data structures for reduced memory usage
  - realize massive parallelism through simultaneous propagation
  - solution techniques for selectively coupled systems
  - optimal propagation sets via selective coupling
  - embedded optimization and UQ capabilities will enable confident predictions of new application areas
- Moreover, such embedded approaches will facilitate the acceleration, and thus, reduce overall complexity of (linear and nonlinear) solvers by exploring multilevel methods and exploiting the inherent hierarchies
- Lots of challenges:
  - significant effort to refactor simulation codes
  - solvers/preconditioners must be optimized for embedded approaches
  - memory access patterns will become critical
  - propagating samples together requires commonality in solution process . . .